# Using Modular Forms for Random Number Generation

Philipp Harland

April 2025

**Abstract**

*In this paper, we will go over a novel method to create pseudorandom number generation, this specific algorithm using modular forms to achieve that goal.*

## Contents

**1 Theory** **1**

**2 Building The Algorithm** **2**

**3 On computation** **3**

## 1 Theory

*A modular form is a specific type of function on the upper half-plane, $\mathcal{H}_{\mathbb{C}}$. An example of a modular form is the Eisenstein series, $\mathbb{E}_{2k}$ for $k \in \mathbb{Z}^{+}$. Formally, it is defined w.r.t. a subgroup $\Omega \subset SL_2(\mathbb{Z})$. The details of this definition can be found on Wikipedia, or in elementary lectures about modular forms. It would be redundant for this paper to discuss the details.*
*We can exploit a specific property of modular forms for cryptography. We have a lattice, $\Lambda$, and a modular form, $M$ – and if we input each of its points that are $\in \mathcal{H}$ into $M$, we will have a lot of variation between given lattice points, even ones that are close together, i.e. separated by at most 1 of either the basis vectors. Some examples are $G_{16}$, and $j(z)$, of which are sketched out in Fig. 1. and Fig. 2.*
*We can use sampling parameter variations from a parameter space, $\mathcal{F}$ to accomplish our goal, of which we will demonstrate in the following chapter. In Fig.1. and Fig.2., you can see the variations in their values w.r.t. perturbations that are smaller in norm than the elements of $\{1, i, 1+i\}$, j(z) especially of the 2.*

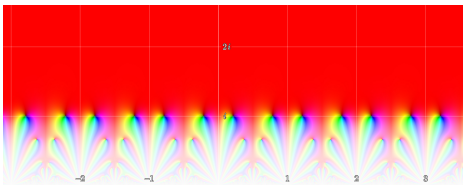

Figure 1: $G_{16}$, *diagram generated by Samuel J. Li's Complex Function Plotter*

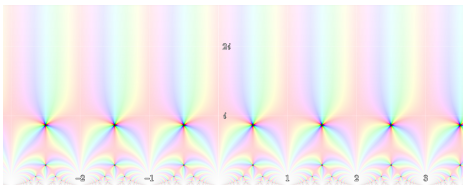

Figure 2: $j(z)$

## 2 Building The Algorithm

*We can "entirefy" our modular form, M, by mapping it to the piecewise function:* $M'(z) = \begin{cases} z \in \mathcal{H} : M(z) \\ z \notin \mathcal{H} : 0 \end{cases}$.
*Now, we can select any complex number, and input it into our function, instead of being restricted to just* $\mathcal{H}$.
*Select a family of modular forms,* $\mathbb{G}$ *which can be moved through by a positive integer parameter* $\pi$ *reigning from a finitely large set* $\Pi$ *in the parameter space* $\mathbb{P}$. *We have the parameter space* $\mathbb{R}\times\mathbb{R}\times\mathbb{C}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\mathbb{R}\times\Pi$. *The first 2 copies of* $\mathbb{R}$ *are for translation within the function's bounds, the middle copy of* $\mathbb{C}$ *is for scaling the input by a complex number, the second to last 2 copies of* $\mathbb{R}$ *are for complex number selection, the last 2 copies are for constant addition, and the set of positive integers* $\Pi$ *is for moving between modular forms (given they have a function between themselves and elements of* $\Pi$*). We have an element,* $(x_1, x_2, w, x_3, x_4, x_5, x_6, \pi)$, *of which we map to* $M'_\pi(w(x_3 + x_4 i) + x_1 + x_2 i) + x_5 + x_6 i$. *Now, we randomly combine a sum of these functions as follows:* $J(z) = \sum_{i=1,\ X_i \in \mathbb{R}_{andom}(\mathbb{P}),}^{n} X_i$. *Now, upon request, i.e. our algorithm is ran once, we will sample a random number, f, and use it as an input in J.*
*This is* ***one*** *way to do it, but we may also introduce other mathematical objects like elliptic curves, or perhaps consider large compositions of modular forms – e.g. compositions of Eisenstein series with intermediary functions being entire functions* $\mathbb{C} \to \mathcal{H}$.
*There are more simplistic ways to generate chaotic pseudorandom numbers using modular forms, such as randomly sampling a single complex parameter with a positive imaginary component to take as an input into a modular form. However, this may be less secure due to local isotropies of the form's value, which are*

*commonly seen in modular forms like the Dirichlet eta function.*

# 3 On computation

*Because we cannot compute infinite precision on a modular form, we are forced to use approximations that terminate in a constant finite number of terms. This is because computers are physical, and only have a finite number of physical bits to store information with. A way to compute modular forms is by their Fourier series, of which there are more details here: Notes by Rich Shwartz, published to the library of Brown University.*
[1]

[1] This paper has been (twice as of 19/4/25) revised to hopefully fit within the standard of the Cryptology ePrint Archive.